

Assignment

by Daniel Samuel

FILE	7148_DANIEL_SAMUEL_ASSIGNMENT_164031_1828864264.DOCX		
TIME SUBMITTED	21-DEC-2018 11:48PM (UTC+0000)	WORD COUNT	2700
SUBMISSION ID	97929726	CHARACTER COUNT	16976

LEVEL 5 ASSESSMENT SPECIFICATION

Student name:	Daniel Samuel	Student ID number:	DS136996
Programme:	Applied Computing		
Module:	Application Development		
Module code:	DUM5002	Contribution to Overall Module Assessment (%):	50%
Lecturer:	Daniel Davies	Internal Verifier:	Denise Hudson
Assignment Title:	Design patterns	Word count (or equivalent):	2500 Words
Submission deadline:	December 21 st 2018	Return date of provisional marks & written feedback:	January 11 th 2019
Submission method:	All written assessments, where practical and possible, must be submitted via Turnitin unless otherwise instructed by the Lecturer. (Please DO NOT put this assessment specification into Turnitin or it will match many similarities with other students' submissions.) Alternative submission method (if applicable): Late submission of the assessment will result in a late penalty mark. Penalties for late submission: Up to one week late, maximum mark of 40%. Over one week late, 0%. Only the Extenuating Circumstances Panel may approve a change to submission dates.		
Academic honesty / referencing:	Academic honesty is required. In the main body of your submission you must give credit to authors on whose research and ideas your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have used, cited or quoted in order to complete this assessment.		

Module Learning Outcomes (from module syllabus)

- 1 - Evaluate the concepts, principles and technologies and techniques used in the development and creation of interactive experiences
- 2 - Select and deploy appropriate scripting and programming languages and techniques in the implementation of applications for real-world problem scenarios.



DANIEL SAMUEL
P149482
APPLICATION DEVELOPMENT
DESIGN PATTERNS
DANIEL DAVIES



Table of Contents

Introduction.....	3
Part A (Understanding).....	3
What are design patterns and why are they used within software development?.....	3
The Gang of four	3
MVC (Mode – View – Controller) pattern	4
What is MVC?.....	4
MVC from an Android development's perspective.....	4
Façade Pattern.....	4
What is the Façade pattern?.....	4
MVVM - (Model - View – View model) pattern	5
MVVM from an Android development's perspective	5
User Experience (UX) design	5
Part B (Implantation).....	6
Dice Rolling App.....	6
CSG Soundboard	6
Text Adventure Game	6
Application Testing	6
Dice Rolling application.....	6
CSG Soundboard application.....	7
Adventure Game	7
What feedback was given?	7
Testing methods that wasn't used.....	7
Conclusion	8
Bibliography	9
Appendices	10
Appendix A	10
Appendix B	11
Appendix C	12

Introduction

In the upcoming report, software patterns / design patterns will be discussed and evaluated on how they could be used for the development of mobile applications. It will also provide a thorough breakdown of the advantages and disadvantages of two different architectural or design patterns and whether they suit being used within the development of applications for mobile devices (such as for Android and iOS devices).

Part A (Understanding)

For this report, the two architectural patterns that will be discussed will be the Façade design pattern and the MVC (Model - View - Controller) architectural pattern. These two patterns were chosen in particular because they are widely used in the software development industry and there are many people who have experience online giving their thoughts on the patterns and whether they think other developers should implement them into their workflow when creating software.

What are design patterns and why are they used within software development?

Design patterns are ways of structuring your code when creating an application on any platform (Android, iOS, Web, etc.). Comparing two definitions of two authors notably Barbara Liskov describes a design pattern as “Organized programs”. In Erich Gamma’s book, he describes design patterns plans. He spoke about a quote given by architect Christopher Alexander which he says: *“Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”*. Both definitions are very different, but both can be seen as true and valid.

The Gang of four

The Gang of Four are a group of people (four computer scientists) who wrote a book called Design Patterns: Elements of Reusable Object-Oriented Software

[1] which became one of the best books people looked too to learn about the various design patterns. According to the Gang of Four, a design pattern has four different parts:

1. A name – is used to summarise the issue at hand, the solution to the problem and also the consequences of the pattern all with as little words as possible.
2. A problem
3. A solution
4. The consequences

They stated in their book that are 3 different types of design patterns, these are:

1. Creational Patterns– Instantiation of objects.
2. Structural Patterns – deal with related objects
3. Behavioural Patterns – Use the communications between objects.

Each design pattern type has many different patterns, some that could be even be associated with more than one meaning that it shares principles from just one design pattern type.

MVC (Mode – View – Controller) pattern

What is MVC?

MVC is an architectural/design pattern for developers to implement user interfaces for their applications [2]. They are most commonly used within web development / web apps but is not necessarily restricted to it. Various companies have used MVC in their own frameworks such as Microsoft. Microsoft use it heavily within their server-side framework called ASP.net [3]. Also, Facebook have an MVC version of their JavaScript UI library called ReactJS.NET which uses part of C# with the already established library [4]. MVC is also a way to organise code into various parts so the code can be flexible, useful and having the ability to be reused in another environment.

Model

The Model of an application is the actual data the view needs to work with. Depending on other aspects of application, the view(s) and controllers(s) will be notified and the necessary actions will be taken if anything on screen should be changed. The model could also be where the database is located if the application requires it.

View

A View is what the user sees on the screen of their device (the user interface). With Android development, the view would include all the images, text components and buttons.

Controller

The controller of an application is where the Model and the View are connected. If the view tells the controller class that user has clicked on an image to enlarge it, the controller will decide what action should be executed.

MVC from an Android development's perspective

In reality, a developer can use any method/pattern they desire to create an Android application. But using a pattern like MVC would be advised because it's been tested, tried and is trusted within the software development industry. When a company feels the need to create an app to offer their services on a new platform, then it's a good opportunity to start creating clean and reliable code. This is where MVC comes in.

According to Eric Maxwell, Android developers for the most part has moved away from design patterns like MVC due to more "modular, testable patterns" being available [5].

Façade Pattern

What is the Façade pattern?

The façade pattern is a very simple design pattern. It allows a developer to split up code very thoroughly into very specific functionality. Doing this will give each bit of code far more room for testability. Using the façade pattern, it allows the developer to create a façade class which hides the complexity of the app behind it and then can access it via an interface object.

The Dummies book for Java programming [6], Doug Lowe describes the façade pattern to have the ability to group common functionality together.

MVVM - (Model - View – View model) pattern

The MVVM pattern is an architectural pattern designed by Ken Cooper while at Microsoft [7]. The main purpose of the pattern is separations of concern, meaning that every part has its own role to play. For example, part 1 could focus on the database and part 2 could focus on the user interface. If this is implemented correctly, then it can greatly improve the readability of the code.

Having parts of the project separated not only technically in code but between members of the team. Doing this can allow people to work on areas of the project that they specialise in. For example, a graphics designer and focus on the user interface.

Model – essentially the data of the application, this is more commonly a database.

View – Like already mentioned in the MVC section, the view of an application is what the end user sees on their screens (UI). This can be manipulated and changed by the model and the view model.

View Model – This layer lives between the model and the view of an application. The view model will process the data from the model and update the view if anything has changed.

MVVM from an Android development's perspective

Eric Maxwell also discusses MVVM with Android Development. He describes it as having the benefits of: “easier testing and modularity, while also reducing the amount of glue code that we have to write to connect the view + model.”.

According to Tian Lou, MVVM is good for designers so if an organisation is working to create an app with a fairly large team. The designers would be able to contribute and not just create the story boards or graphics for the application. Developing an application is difficult enough starting at Hello World but when you add more activities, connect to a database or connect with a JSON API to fetch, the complexity keeps getting more.

User Experience (UX) design

UX design is key to any piece of software in modern day computing. To attract more customers, having a solid UX is key. When someone clicks a button, they expect that button to do something immediately or they'll think that the app is broken. If this happens enough times, then they'll go and find an alternative application to solve the issue they are having. An app needs to be fast and responsive.

On the Android Developer documentation, they suggest a few rules that developers can follow to give users a good user experience [8]. The user could partially use the app without needing to create an account and still get a feel of the functionality and the benefits for them using the app for whatever task it is meant to help with. Something else developers could do to enhance a user's experience is to charge money for the apps (subscription or one-off payment) instead of allowing advertisement. Some people don't mind apps if the app doesn't cost anything but it's when the ads become intrusive and people start to become annoyed and start to think about looking for an alternative to the app.

Part B (Implantation)

A series of very diverse applications have been created to show off different various functionality that can be programmed on Android. There are many methods of creating applications that could have been used but Android Studio and Java was used for these applications. For these apps

Dice Rolling App

The aim for this app is to have two dice, one die belonging to the player and the other belonging to the computer. The user clicks the button and a random number is assigned to the dice and the corresponding number on the face will replace the old number.

CSG Soundboard

The aim for this app is to have a grid of buttons with a single sound assigned to a single button. The user should be able to click a series of buttons to create a tune of their choosing.

Text Adventure Game

The aim of this app is to allow the user / player to select different options to click and depending on what options are pressed, different responses will appear in the set of options.

Application Testing

A series of tests will be carried out to demonstrate that the apps function to a high standard. Alongside with written tests, a number of screenshots will be provided as visual evidence to show that the features do in fact work. A few people have been asked to test the applications to see how satisfied they are in the overall finish of the apps, this is a technique called black box testing which is simply have someone test a piece of software without any knowledge of the workings of the applications code.

For an application to be seen as successful, it's important that it can run on various devices and still look the same. A series of tests will be performed to see how the apps look on various devices (using the built-in android device emulators in Android Studio).

When writing an application that is intended to be released into the play store, testing the app on different versions of Android is important. This is because the older the version of Android that your app supports and can run on, the more potential people will download your app. According to Digital Trends, Oreo runs on only 19% of Android devices [9]. That number is not what should be aimed at,

Dice Rolling application

In the appendices (Appendix B) at the end of the report, you will see a series of screenshots of the dice rolling app running on 2 different devices, notably the Nexus 6 & Pixel 2 both running API version 28. These devices were chosen in particular as they were released a few years apart from each other and are also different screen sizes. The later point was a particularly important thing to keep in mind when choosing which devices to run the tests on because it demonstrates exactly how well the constraints system works within Android Studio and the Android operating system.

CSG Soundboard application

For this application two different device emulators were chosen and also from different companies (non Google devices). This was done because in the previous app, both devices were ran using emulators of Google own made devices so running on other phone manufactures devices show it's not just an app working well made with Software made by Google running on a phone made by Google.

Disclaimer: Android Studio does not have any third party pre made phone emulators available so one of the screenshots was created from a new hardware profile and the device was adapted to be like a Samsung galaxy phone.

Screenshots of the application running can be found in appendix A below.

Adventure Game

For the purpose of testing the applications created a number of people were asked (black box testers) to test the apps, in which were first installed onto a Samsung Galaxy S9 running Android version 9.0 Pie.

Screenshots of the application running can be found in appendix C below.

What feedback was given?

Dice Rolling app

Initially, the feedback that was given shown that the apps were freezing every time the app would be loaded on the Samsung Galaxy s9, but a simple reinstallation of the app solved the issue. When the app loaded successfully for the first time, the app worked as was hoped when in development. The button was pressed and then the two Dice were rolled, and a winner was decided from the two outcomes.

Originally the game app did not have a scoring system, but it was suggested and was then implemented into the game. Having this allows the user to look at app more like a game rather an app which rolls two dice.

Testing methods that wasn't used

Junit

Junit is a Java unit testing framework. It was created by Erich Gamma and Kent Beck. It's a framework which isn't made specifically for testing on Android but for non-android java-based applications.

Espresso

Espresso is testing framework to test the user interface [10]. Espresso was not used because the user interfaces for the three applications created are very simplistic and were also constructed using constraints which are generally reliable and dependable.

Conclusion

In conclusion, design patterns seem to be far more useful than that was thought prior to the investigation and research. After researching three different patterns, it is clear now that design patterns will be used in the future when developing. After creating the

Bibliography

- [1] R. H. R. J. J. V. Erich Gamma, Design Patterns: Elements of Reusable Object-Oriented Software, Upper Saddle River, New Jersey: Addison Wesley, 1994.
- [2] Mozilla Developer Network, "MVC architecture," Mozilla, 8th June 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture. [Accessed 27th November 2018].
- [3] Rick Anderson, "Getting started with ASP.NET with MVC 5," Microsoft, 10th April 2018. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started>. [Accessed 2nd December 2018].
- [4] Facebook, "ReactJS.Net," Facebook, 2014. [Online]. Available: <https://reactjs.net/>. [Accessed 13th December 2018].
- [5] E. Maxwell, "MVC vs. MVP vs. MVVM on Android," Realm, 26th January 2017. [Online]. Available: <https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/>. [Accessed 27th November 2018].
- [6] D. Lowe, Java All In One For Dummies, vol. 4, Hoboken, New Jersey: John Wiley & Sons, 2014, p. 481.
- [7] K. Cooper, "Projects," Ken Cooper, [Online]. Available: <http://coopercode.com/>. [Accessed 30th November 2018].
- [8] Google and Android, "UX best practices for apps on Google Play Instant," Google and Android, 9th August 2018. [Online]. Available: <https://developer.android.com/topic/google-play-instant/best-practices/apps>. [Accessed 15th December 2018].
- [9] B. S. a. J. Chokkattu, "Android 8.0 Oreo operating system is now on 19 percent of active devices," Digital Trends, 24th October 2018. [Online]. Available: <https://www.digitaltrends.com/mobile/android-distribution-news/>. [Accessed 15th December 2018].
- [10] G. a. Android, "Espresso," Google and Android, 7th August 2018. [Online]. Available: <https://developer.android.com/training/testing/espresso/>. [Accessed 16th December 2018].

Appendices

Appendix A

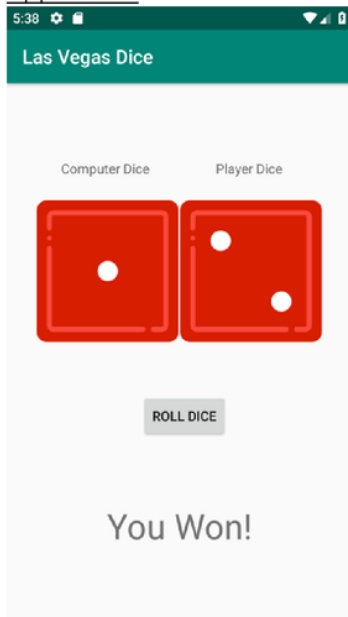


Nexus 6 with the sound board app

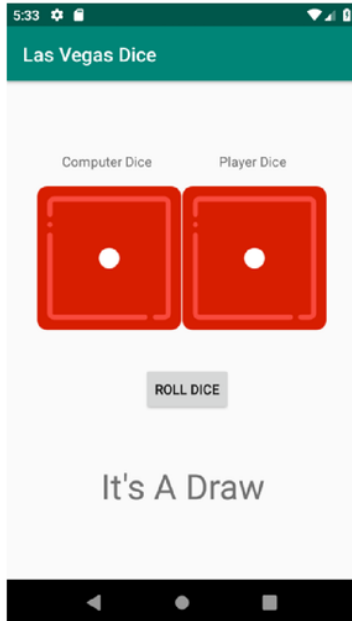


Pixel 2 running the sound board app

Appendix B

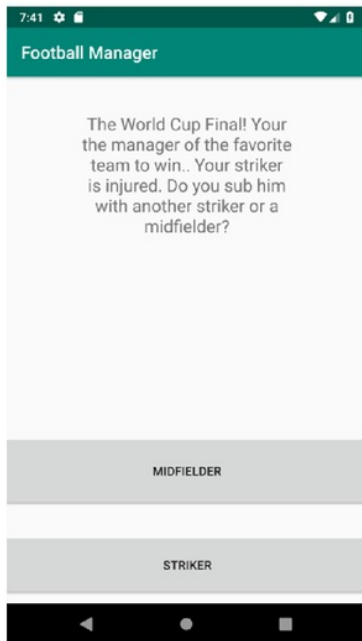


Custom made emulator running the Dice rolling app.

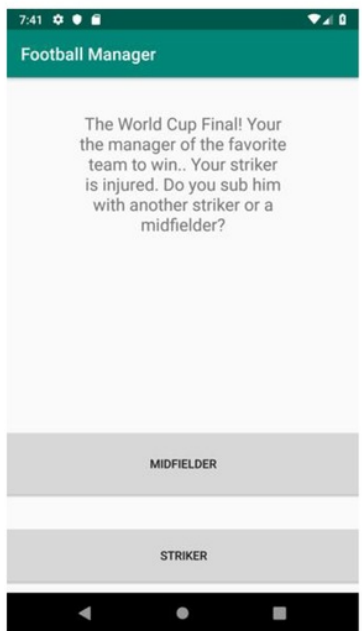


Pixel 2 running the sound Dice rolling app

Appendix C



Nexus 6 running text adventure game



Pixel 2 running text adventure game